

BEAN(Business English Activity lessoN)

: 질문하고 말하며 배우는 영어 클래스!

파워레인조

성아름(팀장), 김선아, 배유진, 조용승

목차

회원가입 /로그인 기능 (kakao LoginAPI, naver LoginAPI, 일반 로그인), 프로필 확인

Spring Security

카카오 지도 API

결제 및 수업/스케줄 관리 (결제 API, Spring Quartz, smtp)

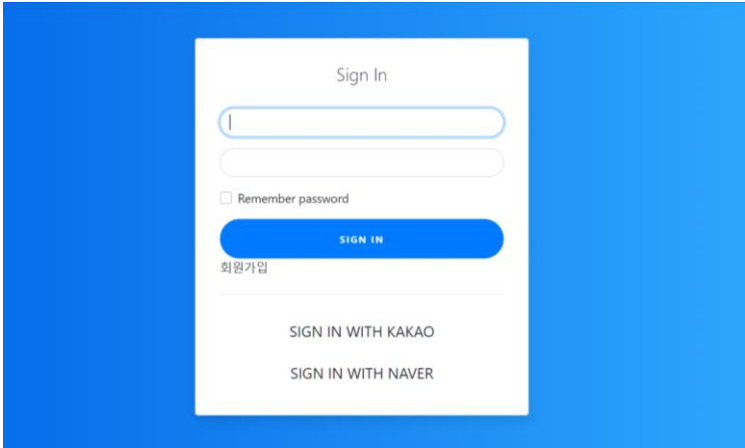
질문 게시판(flask 통신, OCR)

번역 기능(PaPago API)

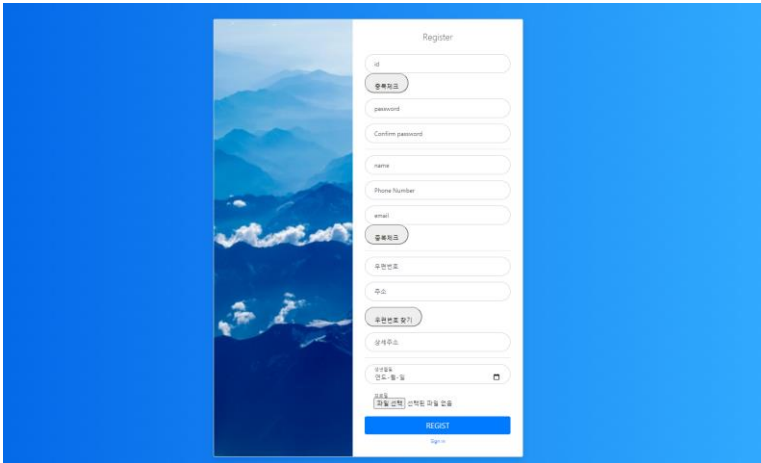
실시간 채팅

리뷰 게시판(검색, 페이징)

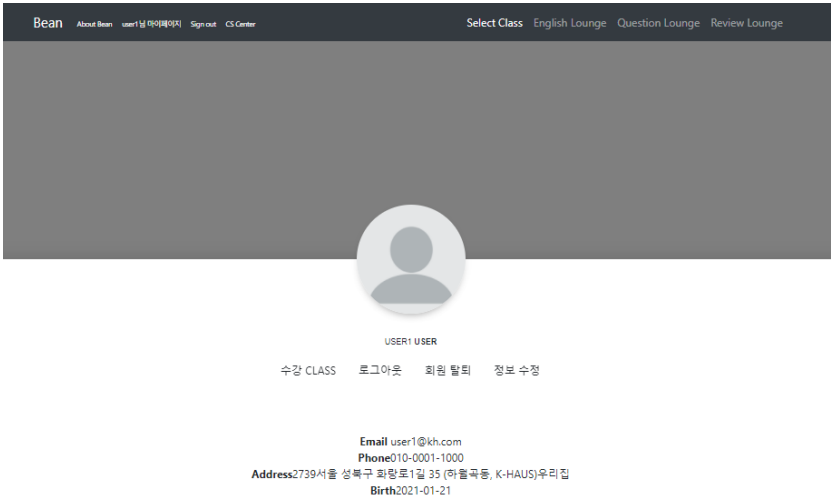
회원가입 /로그인 기능 (kakao LoginAPI, naver LoginAPI, 일반 로그인), 프로필 확인



- 일반 아이디/네이버/카카오에 가입된 아이디와 비밀번호를 통해 로그인 할 수 있다.



- Bean에 회원가입을 진행할 수 있다



- 마이 페이지에서 프로필을 확인할 수 있다

LoginController_Bean.java

```
// 회원가입
@RequestMapping(value = "/resister.do", method = RequestMethod.POST)
public String register(MultipartHttpServletRequest request, Model model, LoginDto dto,
    @RequestParam("member_mpfile") MultipartFile file, BindingResult result) {

    boolean filechk = true;
    if (file.getSize() == 0) {
        filechk = false;
    }

    if (filechk) {
        String name = "";
        String oldname = file.getOriginalFilename();
        int index = oldname.lastIndexOf(".");

        if (index != -1) { // 파일 확장자 위치
            name = date2 + oldname.substring(0, index) + time2 +
                oldname.substring(index, oldname.length()); // 현재
        }
        dto.setMember_imgname(name);

        InputStream inputStream = null;
        OutputStream outputStream = null;

        try {
            inputStream = file.getInputStream(); // 파일내용을 읽기 위해 inputStream을 받아옴
            String path = WebUtils.getRealPath(request.getSession().getServletContext(),
                "/resources/storage");

            File storage = new File(path);
            if (!storage.exists()) {
                storage.mkdirs();
            }

            File newFile = new File(path + "/" + name);

            if (!newFile.exists()) {
                newFile.createNewFile();
            }

            dto.setMember_imgpath(path);

            outputStream = new FileOutputStream(newFile);

            int read = 0;
            // 읽은 데이터 크기
            byte[] b = new byte[(int) file.getSize()];

            while ((read = inputStream.read(b)) != -1) {
                outputStream.write(b, 0, read);
            }

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                inputStream.close();
                outputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
dto.setMember_pw(passwordEncoder.encode(dto.getMember_pw()));
dto.setMember_pwchk(passwordEncoder.encode(dto.getMember_pwchk()));

int res = biz.resister(dto);

if (res > 0) {
    return "mainpage";
}

return "registBean";
}
```

LoginController_Bean.java

- file 객체의 크기를 확인해서 만약에 크기가 0이라면
- 비밀번호 암호화 진행 후 저장

login.jsp

```
<script type="text/javascript">
    var naver_id_login = new naver_id_login("",
        "http://localhost:8787/bean/navercallback.do");
    var state = naver_id_login.getUniqState();
    naver_id_login.setButton("green", 3, 45);
    naver_id_login.setDomain("http://localhost:8787/bean");
    naver_id_login.setState(state);
    naver_id_login.init_naver_id_login();
</script>
```

navercallback.jsp

```
<script type="text/javascript">
    var naver_id_login = new naver_id_login("",
        "http://localhost:8787/bean/navercallback.jsp");
    var token = naver_id_login.oauthParams.access_token;
    naver_id_login.get_naver_userprofile("naverSignInCallback()");

    function naverSignInCallback() {
        var name = naver_id_login.getProfileData('name');
        var id = naver_id_login.getProfileData('id');
        var email = naver_id_login.getProfileData('email');

        var data = {
            "member_name" : name,
            "member_sns" : id,
            "member_email" : email
        }

        $.ajax({
            url : "naverlogin.do",
            type : "POST",
            data : JSON.stringify(data),
            contentType : "application/json",
            dataType : "json",
            success : function(msg) {
                console.log(msg);
                if (msg.check == true) {
                    location.href = "index.jsp";
                } else {
                    location.href = "naverform.do";
                }
            },
            error : function(request, status, error) {
                alert("통신 실패");
                alert("code = " + request.status + " message = "
                    + request.responseText + " error = " + error);
            }
        });
    }
</script>
```

LoginController_Naver

```
// 네이버 로그인 완료 후 정보 받아서 처리할 페이지
@RequestMapping(value = "/naverlogin.do", method = RequestMethod.POST)
@ResponseBody
public Map<String, Boolean> naverlogin(@RequestBody LoginDto dto , HttpSession session) {

    String member_sns = "naver" + dto.getMember_sns();

    LoginDto res = biz.snsChk(member_sns);
    boolean check = false;

    if (res != null) {

        session.setAttribute("login", res);
        check = true;

    } else {

        LoginDto add = new LoginDto();
        add.setMember_sns(member_sns);
        add.setMember_email(dto.getMember_email());
        add.setMember_name(dto.getMember_name());
        session.setAttribute("info", add);

    }

    Map<String, Boolean> map = new HashMap<String, Boolean>();
    map.put("check", check);

    return map;
}
```

login.jsp

- login.jsp 를 통해 네이버 로그인 서비스 창으로 이
- 네이버 로그인 후 해당 유저의 접근 토큰 값이 콜백 페이지로 넘어옴

navercallback.jsp

- 콜백 페이지에 넘어온 유저의 정보를 ajax로 컨트롤러 페이지로 넘겨줌
- 가입 이력이 있다면 컨트롤러에서 넘어온 정보를 가지고 메인 페이지로 이동
- 가입 이력이 없다면 회원 가입용 페이지 이동

LoginController_Naver

- 넘겨준 정보를 db와 비교
- 가입 된 이력이 있다면 "login"을 세션 객체에 정보를 담아 줌
- 가입 된 이력이 없다면 "info"를 세션 객체에 담아 줌

LoginController_Kakao

```

@RequestMapping(value = "/kakaoLogin.do", method = { RequestMethod.GET, RequestMethod.POST })
public String kakaoLogin(@RequestParam("code") String code, HttpServletRequest request,
    HttpServletResponse response, HttpSession session) throws Exception {

    LoginDto add = null;

    JsonNode token = KakaoLogin.getAccessToken(code);

    JsonNode profile = KakaoLogin.getKakaoUserInfo(token.path("access_token").toString());
    LoginDto dto = KakaoLogin.changeData(profile);

    dto.setMember_name(dto.getMember_name());
    dto.setMember_email(dto.getMember_email());
    dto.setMember_sns(dto.getMember_sns());

    LoginDto res = new LoginDto();
    res = biz.snsChk("kakao" + dto.getMember_sns());

    if (res != null) {
        session.setAttribute("login", res);
        return "mainpage";
    }

    add = new LoginDto();

    add.setMember_sns(dto.getMember_sns());
    add.setMember_email(dto.getMember_email());
    add.setMember_name(dto.getMember_name());
    session.setAttribute("info", add);

    return "registKakao";
}

```

KaKaoLogin.java

```

public static JsonNode getAccessToken(String authorize_code) {
    final String RequestUrl = "https://kauth.kakao.com/oauth/token";

    final List<NameValuePair> postParams = new ArrayList<NameValuePair>();
    postParams.add(new BasicNameValuePair("grant_type", "authorization_code"));
    postParams.add(new BasicNameValuePair("client_id", KAKAO_REST_API_KEY)); // REST API KEY
    postParams.add(new BasicNameValuePair("redirect_uri", "http://localhost:8787/bean/kakaoLogin.do")); // 리다이렉트 URI
    postParams.add(new BasicNameValuePair("code", authorize_code)); // 로그인 과정에서 얻은 code 값

    final HttpClient client = HttpClientBuilder.create().build();
    final HttpPost post = new HttpPost(RequestUrl);
    JsonNode returnNode = null;

    try {
        post.setEntity(new UrlEncodedFormEntity(postParams));
        final HttpResponse response = client.execute(post);
        final int responseCode = response.getStatusLine().getStatusCode();

        // JSON 형태 반환값 처리
        ObjectMapper mapper = new ObjectMapper();
        returnNode = mapper.readTree(response.getEntity().getContent());

    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (ClientProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {}

    return returnNode;
}

public static JsonNode getKakaoUserInfo(String authorize_code) {

    final String RequestUrl = "https://kapi.kakao.com/v2/user/me";

    final HttpClient client = HttpClientBuilder.create().build();
    final HttpPost post = new HttpPost(RequestUrl);

    // add header
    post.addHeader("Authorization", "Bearer " + authorize_code);

    JsonNode returnNode = null;

    try {
        final HttpResponse response = client.execute(post);
        final int responseCode = response.getStatusLine().getStatusCode();

        System.out.println("\nSending 'POST' request to URL : " + RequestUrl);
        System.out.println("Response Code : " + responseCode);

        // JSON 형태 반환값 처리
        ObjectMapper mapper = new ObjectMapper();
        returnNode = mapper.readTree(response.getEntity().getContent());

    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (ClientProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        // clear resources
    }

    return returnNode;
}

```

LoginController_Kakao

- 카카오톡을 이용해 회원가입 및 로그인 진행

KaKaoLogin.java

- 카카오톡에서 토큰 값 및 유저 정보를 받아 옴

```

<form class="form-signin" action="/bean/Logininfo.do" method="post">
  <div class="form-label-group">
    <input type="text" name="member_id" placeholder="id" class="form-control"
      required autofocus>
  </div>
  <div class="form-label-group">
    <input type="password" name="member_pw" placeholder="password" class="form-control"
      placeholder="Password" required>
  </div>

  <div class="custom-control custom-checkbox mb-3">
    <input type="checkbox" class="custom-control-input"
      id="customCheck1"> <label class="custom-control-label"
        for="customCheck1">Remember password</label>
  </div>
  <button class="btn btn-lg btn-primary btn-block text-uppercase"
    type="submit">Sign in</button>
</form>

```

LoginController_Bean.java

```

@RequestMapping("/login.do")
public String login(HttpSession session, Model model) {

    Authentication auth = SecurityContextHolder.getContext().getAuthentication();
    String member_id = auth.getName();
    LoginDto res = biz.login(member_id);

    if (res != null) {
        if (res.getMember_withdrawal().equals("N")) {
            session.setAttribute("login", res);
            return "mainpage";
        } else if (res.getMember_withdrawal().equals("Y")) {
            String msg = "탈퇴한 회원입니다.";
            model.addAttribute("msg", msg);
            return "login_withdrawal";
        }
    }

    return "login";
}

```

login.jsp

- login.jsp에서 id와 password를 입력
- 이후 id와 password가 security로 넘어 감

Security

- security에서 로그인 후 컨트롤러로 이동

LoginController_Bean.java

- security에서 정보를 가져와 아이디로 유저의 정보를 찾음
- 탈퇴한 회원이 아닐 경우 세션 객체에 값을 넣어 메인 페이지로 넘겨 줌

<느낀 점>

수업시간에 자주 해봐서 일반 로그인은 어렵지 않았다. API를 이용해서 로그인 하는것도 컨트롤러를 통해서 값을 가져오는건 어렵지 않았는데, API와 연결하는게 좀 어려웠던것 같다.

mypage_profile.jsp

```
<script type="text/javascript"
src="https://code.jquery.com/jquery-3.4.1.min.js"></script>

<script type="text/javascript">
$(document).ready(function(){
$.ajax({
type : "post",
url : "profileimg.do",
contentType : "application/json",
dataType : "json",
success : function(msg){
console.log(msg);
if(msg.check == true){
$("#profileimg").attr("src", "${pageContext.request.contextPath }/resources/storage/${login.member_imgname}");
} else {
$("#profileimg").attr("src", "resources/images/profile/profile.png");
}
}, error : function(){
alert("통신 실패");
}
});
});
</script>
```

MemberController.java

```
// 프로필 화면의 사진을 로딩하는 컨트롤러
@RequestMapping(value = "/profileimg.do", method = RequestMethod.POST)
@ResponseBody
public Map<String, Boolean> displayPhoto (HttpServletRequest response, Model model, HttpSession session) throws Exception {

ServletOutputStream bout = response.getOutputStream();

LoginDto dto = (LoginDto) session.getAttribute("login");
String imgpath = dto.getMember_imgpath() + "\\\" + dto.getMember_imgname();

int index = dto.getMember_imgname().lastIndexOf(".");
String file = dto.getMember_imgname().substring(index, dto.getMember_imgname().length());

if(file.equals(".jpg")) {
response.setContentType("image/jpeg");
} else if(file.equals(".png")) {
response.setContentType("image/png");
}

boolean check = false;
Map<String, Boolean> map = new HashMap<String, Boolean>();

//파일의 경로
FileInputStream f;
int length;
byte[] buffer = new byte[10];
try {
f = new FileInputStream(imgpath);
while((length=f.read(buffer)) != -1){
bout.write(buffer,0,length);
}
check = true;
map.put("check", check);
} catch (FileNotFoundException e) {
check = false;
map.put("check", check);
System.out.println("error catch : 파일없음");
}

return map;
}
```

MemberController.java

- js 실행 시 ajax로 컨트롤러에서 파일 경로와 이름을 찾아 파일을 읽을 수 있다면 check = true
- 파일이 없어 예외 발생 시에는 check= false

mypage_profile.jsp

- check=true일 경우, 해당 이미지를 사용할 수 있으므로 src 옵션에 경로와 값을 response
- check = false일 경우 resources/images/profile에 들어있는 디폴트 이미지인 profile.png를 사용

<느낀 점>

파일 경로 잡아서 가져오는게 힘들었다.
서버를 내리거나 클린해서 서버에 이미지가 사라졌을 경우 경로 값은 저장되어 있으나 파일을 불러오지 못하는 문제가 발생했다. 이를 보완하는 것에 주안점을 두고 코드를 작성했다

SpringSecurity

security를 통해 사용자 계정의 정보를 암호화하고, 각 url마다의 접근권한을 설정해준다.

Spring Security

Security-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
  xmlns:beans="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/security
    http://www.springframework.org/schema/security/spring-security.xsd
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-4.3.xsd">
```

```
<context:component-scan base-package="com.power.security" />
```

① <http auto-config="true" use-expressions="true">

```
<!-- 자원 관련 url -->
<intercept-url pattern="/resources/**" access="permitAll" />

<!-- main page 관련 url -->
<intercept-url pattern="/index.jsp" access="permitAll" />
<intercept-url pattern="/header.jsp" access="permitAll" />
<intercept-url pattern="/main.do" access="permitAll" />
<intercept-url pattern="/mainpage.jsp" access="permitAll" />
<intercept-url pattern="/intro.do" access="permitAll" />
```

②

```
<!-- 로그인/회원가입 관련 url -->
<intercept-url pattern="/Loginform.do" access="permitAll" />
<intercept-url pattern="/registtype.do" access="permitAll" />
<intercept-url pattern="/resister.do" access="permitAll" />
<intercept-url pattern="/idChk.do" access="permitAll" />
<intercept-url pattern="/emailChk.do" access="permitAll" />
<intercept-url pattern="/registform.do" access="permitAll" />
<intercept-url pattern="/Logininfo.do" access="permitAll" />
<intercept-url pattern="/**" access="permitAll" />
```

③

```
<form-login login-page="/Loginform.do"
  login-processing-url="/Logininfo.do"
  default-target-url="/Login.do"
  username-parameter="member_id"
  password-parameter="member_pw"
  authentication-failure-url="/Login.do?error=1"
  always-use-default-target="true" />
```

pom.xml

- Spring security version 을 4.2.19.RELEASE 로 설정
- spring-security-core(인증 인가 기능), spring-security-web(보안 기능, pw encoding/decoding), spring-security-config, spring-security-taglibs를 pom.xml에 추가

servelt-context.xml / web.xml

- springSecurityFilterChain 설정(org.springframework.web.filter.DelegatingFilterProxy)

Security-context.xml

- security-context : 실제 spring security가 작동되는 부분

① <http>

- auto-config='true' : 기본 로그인페이지/ HTTP 기본인증 / 로그아웃기능 등을 제공
- use-expressions="true" : SpEL 을 사용한다는 의미
- * SpEL 은 intercept-url에서 access의 표현식이며, 코드상의 permitAll이 여기에 속함

② <intercept-url>

- intercept-url의 권한은 위쪽이 우선시 됨
- pattern : 접근하려는 url pattern을 의미
- access : 해당 url에 접근 가능한 권한
- * 추가로 접근가능한 IP 등을 설정 가능
- permitAll : 모든 권한을 허용한다는 뜻
- * hasRole('ROLE_USER') 을 설정해 ROLE_USER라는 권한을 가진 사용자만 해당 url로의 접근을 허용할 수 있다

③ <form-login>

- 로그인 수행에 직접적으로 관여하는 부분
- login-page : 로그인 페이지 url 설정
- login-processing-url : 실제 로그인 폼에서 action으로 이어지는 url
- * spring security에서 만들어진 페이지를 사용하고 싶다면 이 옵션을 추가하지 않고 로그인 폼에 POST 형 태로 url을 /login 이라 적어주면 된다.
- default-target-url : 로그인이 성공 후 이동할 페이지를 설정
- username-parameter / password-parameter : login form 안에 들어간 id/pw name과 동일한 id/pw name으로 설정하여 로그인 시 해당 아이디와 비밀번호를 가져올 수 있게 됨
- authentication-failure-url : 실패시 호출해줄 URL(default : /login?error=1)

Security-context.xml

```
④ <logout invalidate-session="true"
    logout-url="/LogoutSecurity.do" logout-success-url="/Logout.do"
    delete-cookies="true" />
⑤ <csrf disabled="true" />
⑥ <access-denied-handler error-page="/error.do" />
</http>

⑨ <authentication-manager>
    <authentication-provider>
        user-service-ref="securityService">
        <password-encoder ref="passwordEncoder" />
    </authentication-provider>
</authentication-manager>

⑧ <beans:bean id="passwordEncoder"
    class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder" />
</beans:beans>
```

Security-context.xml

④ <logout>

- 로그아웃 수행에 직접적으로 관여하는 부분
- invalidate-session : 로그아웃 시 세션 초기화 여부(true 설정 시 세션 초기화)
- logout-success-url : 로그아웃 시 이동할 페이지 설정 옵션
- logout-url : 로그아웃 시 URL 설정 옵션(default : POST방식의 /logout)
- delete-cookies : 로그아웃시 쿠키 삭제 여부를 말하며 true의 경우 쿠키 삭제가 진행

⑤ <csrf>

- csrf disabled="true"는 csrf를 사용하지 않겠다는 뜻
- spring security는 4.x버전부터 디폴트로 작동

⑥ <access-denied-handler>

- access-denied-handler error-page="/error.do" : 로그인에 되었지만 페이지에 대한 접근 권한을 가지고 있지 않을 경우 처리해줄 url

⑨ <authentication-manager>

- authentication-manager : 로그인 시 해당 아이디와 비밀번호가 db에 있는지를 확인해 로그인 여부를 확인해주기 위한 내용을 넣는 최상위 태그
- authentication-provider 태그 안에 user-service 태그를 넣음
 - * <user name="user" password="userPw" authorities="ROLE_USER"/> 식으로 계정을 생성 가능
 - * user-service-ref="securityService" 옵션을 이용해서 계정을 가져올 수 있음
 - * <password-encoder ref="passwordEncoder" />를 넣어서 인코딩된 pw를 디코딩해줄 수 있음

⑧ passwordEncoder

- BCryptPasswordEncoder를 이용해 비밀번호를 인코딩/디코딩 하기위한 bean 생성

SecurityService

```
@Service
public class SecurityService implements UserDetailsService {

    Logger log = LoggerFactory.getLogger(this.getClass());

    @Autowired
    private SecurityDao dao;

    public SecurityService() {
    }

    @Override
    public UserDetails loadUserByUsername(String member_id) throws UsernameNotFoundException {

        LoginDto dto = new LoginDto();
        dto = dao.login(member_id);

        SecurityUser user = new SecurityUser();

        if(dto == null) {
            throw new UsernameNotFoundException(member_id);
        } else if (dto.getMember_id() == null) {
            throw new UsernameNotFoundException(member_id);
        } else {
            user.setUsername(dto.getMember_id());
            user.setPassword(dto.getMember_pw());

            if(!dto.getMember_type().equals("A")) {
                List<String> list = new ArrayList<String>();
                list.add("ROLE_USER");
                user.setAuthorities(list);
            } else {
                List<String> list = new ArrayList<String>();
                list.add("ROLE_ADMIN");
                user.setAuthorities(list);
            }
        }

        return user;
    }
}
```

SecurityUser

```
public class SecurityUser implements UserDetails {

    private static final long serialVersionUID = 1L;

    private String member_id; // ID
    private String member_pw; // PW
    private List<GrantedAuthority> authorities;

    public void setUsername(String member_id) {
        this.member_id = member_id;
    }

    public void setPassword(String member_pw) {
        this.member_pw = member_pw;
    }

    public void setAuthorities(List<String> authList) {

        List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();
        for (int i = 0; i < authList.size(); i++) {
            authorities.add(new SimpleGrantedAuthority(authList.get(i)));
        }
        this.authorities = authorities;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {

        return authorities;
    }

    @Override
    public String getPassword() {
        return member_pw;
    }

    @Override
    public String getUsername() {
        return member_id;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }
}
```

SecurityDao

- SecurityDao가 로그인한 아이디에 맞는 유저 정보를 가져 오

SecurityService

- security-context에서 계정을 확인할 때 사용
- id를 가지고 유저의 정보를 확인
- 만일 정보가 없다면 UsernameNotFoundException을 발생시킴
- 유저 정보가 있다면 계정 정보 중 아이디와 비밀번호, 회원 타입에 따른 권한을 user객체에 담아 리턴 해 줌

SecurityUser

- SecurityUser은 UserDetails를 상속받아 만든 페이지로, User 객체를 만들기 위해 사용 됨

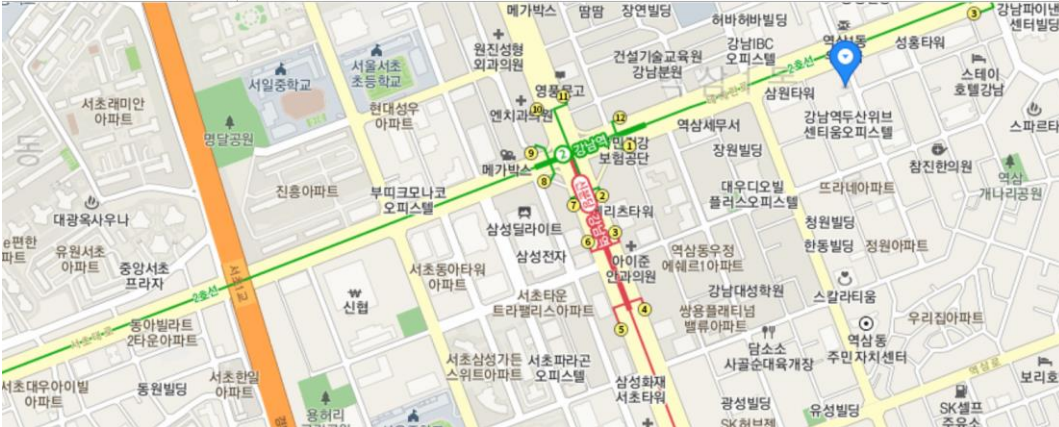
<느낀 점>

- 로그인/회원가입을 이미 만들어놓고 기능을 추가해주었기 때문에 경로를 잡거나 필터를 처리해주는 부분이 어려웠다
- UserDetailsService 상속받은 SecurityService 구현이 어려웠다.

카카오맵 지도

Bean Academy로 오시는 길입니다.

주소	서울특별시 강남구 테헤란로 14길 6 남도빌딩
버스	역삼역.포스코P&S타워 정류장 146 / 740 / 341 / 360 1100 / 1700 / 2000 / 7007 / 8001
지하철	지하철 2호선 역삼역 3번출구 100m



카카오 지도를 이용해 찾아오는 길을 보여준다

bean_intro.jsp

```
<td colspan="2">
    <div id="map" style="width: 100%; height: 450px;"></div>
</td>

<script type="text/javascript">
    var mapContainer = document.getElementById('map'), // 지도를 표시할 div

    mapOption = {
        center : new kakao.maps.LatLng(37.498854411970214,
            127.03333265618686), // 지도의 중심좌표
        level : 2
    }, // 지도의 확대 레벨

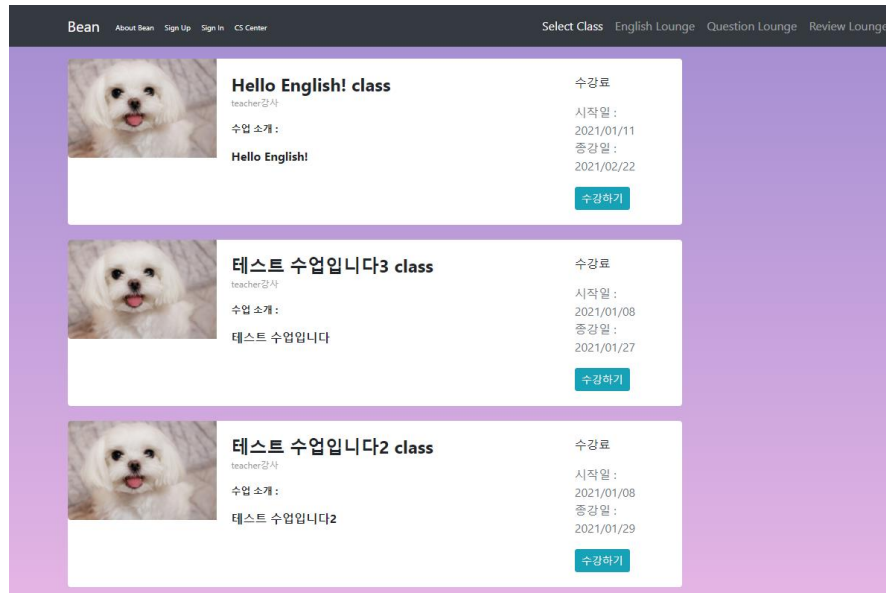
    // 지도를 표시할 div와 지도 옵션으로 지도를 생성합니다
    var map = new kakao.maps.Map(mapContainer, mapOption);

    // 마커가 표시될 위치입니다
    var markerPosition = new kakao.maps.LatLng(37.49900095302427,
        127.03287754168265);

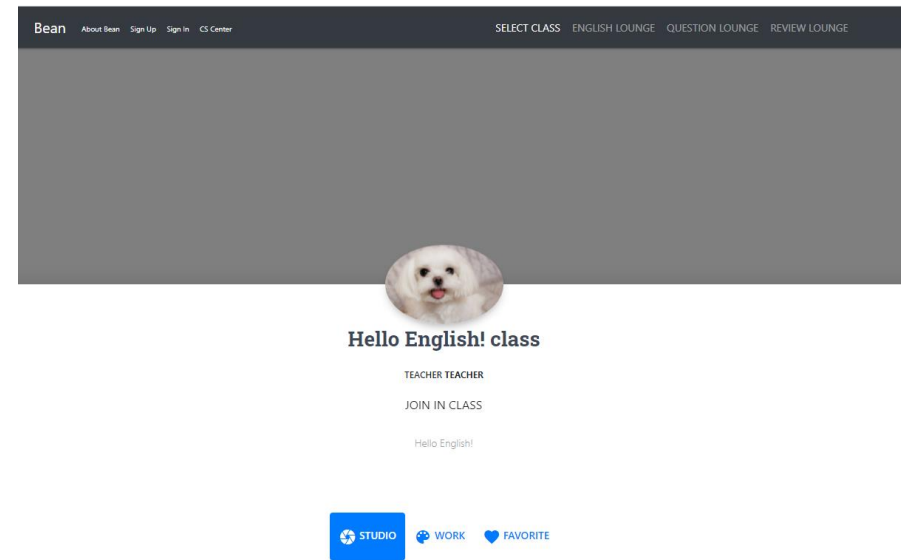
    // 마커를 생성합니다
    var marker = new kakao.maps.Marker({
        position : markerPosition
    });

    // 마커가 지도 위에 표시되도록 설정합니다
    marker.setMap(map);
</script>
```

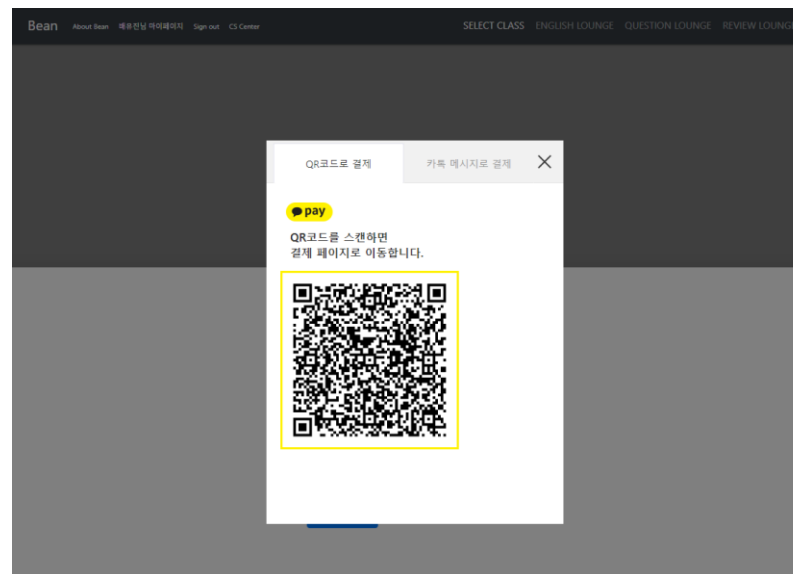

결제 및 수업/스케줄 관리



- 수강 class를 선택할 수 있다



- class 세부사항을 볼 수 있다



- 결제를 진행할 수 있다

class_paying.jsp

```

if (obj['value'] == "true") {
    var jsonData;
    IMP.request_pay({
        pg : 'inicis', // version 1.1.0부터 지원.
        pay_method : 'kakaoPay',
        merchant_uid : 'merchant_' + new Date().getTime(),
        name : '주문영:결제테스트',
        amount : 34000,
        buyer_name : obj["userName"]
    });
    function(rsp) {
        if (rsp.success) {

            //필요한 값을 object 객체에 추가해 줄 것
            remainData.usernum = obj["usernum"]; //buyer 회원 번호

            jsonData = JSON.stringify(remainData);

            $.ajax({
                url : "insertPaying.do",
                method : "POST",
                data : {

                    "jsonData" : jsonData,
                    "member_no" : obj["usernum"],
                    "class_no" : classObj["class_no"],
                    "payment_price" : 34000,
                    "payment_method" : 'KakaoPay',
                    "payment_impuid" : rsp.imp_uid
                },
                dataType : "json",
                success : function(returnData) {

                    if (returnData["check"]) {

                        var msg = "결제에 성공했습니다";
                        alert(msg);

                    } else {

                        var msg = '결제에 실패하였습니다.';
                        msg += '결제가 취소되었습니다'

                    }

                })
            } else {

                var msg = '결제에 실패하였습니다.';
                msg += '에러내용 : ' + rsp.error_msg;
                alert(msg);
            }
        }
    }
}

```

PayingController.java

```

@RequestMapping("/insertPaying.do")
@ResponseBody
public Map<String, Boolean> insertPaying(String jsonData, PayingDto payingDto){

    Boolean check = true;
    PayingDto insertDto = payingDto;
    Map<String, Boolean> map = new HashMap<String, Boolean>();

    // Apache HttpClient기반의 java를 아임포트 REST API클라이언트
    IamportClient iamportClient = new IamportClient();
    String apiResponse;
    ObjectMapper mapper = new ObjectMapper();
    JsonParser parser = new JsonParser();

    Map<String, String> ajaxmap;
    Map<String, String> resultMap;
    Map<String, String> realResponse;

    String realAmount;

    try {
        //실제 결제 된 값이 결제 한 값과 같은 지 확인
        apiResponse = iamportClient.paymentByImpUid(payingDto.getPayment_impuid());
        Object obj = parser.parse(apiResponse);
        JSONObject jsonObj = (JSONObject) obj;
        JsonElement response = jsonObj.get("response");
        realAmount = response.getAsJsonObject().get("amount").getAsString();

        //같은 값을 경우 -> 영수증 데이터에 Dto 추가, class에 회원 추가
        if(realAmount.equals(payingDto.getPayment_price())) {

            insertDto.setPayment_state("Y");
            insertDto.setPayment_refund("N");

            int payingres = payingBiz.insertPaying(insertDto);
            int classes = classBiz.updateClassStudent(insertDto.getClass_no(),
                insertDto.getMember_no(), insertDto.getPayment_impuid());

            if(payingres <= 0 || classes <= 0) {
                check = false;
            } else {

                insertDto.setPayment_state("N");
                insertDto.setPayment_refund("Y");
                int pyaingres = payingBiz.insertPaying(payingDto);

            }

        } catch (Exception e) {
            e.printStackTrace();
        }

        map.put("check", check);

    }

    return map;
}

```

아임포트 서버

IamportClient.java

```

public String getToken() throws Exception {

    IamportResponse<AccessToken> auth = this.getAuth();

    if (auth != null) {
        String token = auth.getResponse().getToken();
        return token;
    }

    return null;
}

public String paymentByMerchantUid(String merchantUid) throws Exception {

    String token = this.getToken();

    if (token != null) {

        String path = "/payments/find/" + merchantUid;
        String response = this.getRequest(path, token);

        Type listType = new TypeToken<IamportResponse<Payment>>() {
        }.getType();
        String payment = gson.fromJson(response, listType);

        return payment;
    }

    return null;
}

```

class_paying.jsp

- IMP 결제 요청
- payingController로 후처리 작업 및 결제/class 정보 저장 요청

PayingController.java

- 전송 받은 결제 정보와 아임포트 서버에 저장된 값을 비교
- 일치 시 class/결제 정보를 Json 형식으로 sql에 저장
- 불일치 시 check=false를 전송

IamportClient.java

- API 값을 이용해 auth 진행
- Auth 이후 token 값 생성
- Token 값을 이용해 후처리 진행
- Apache HttpClient기반 통신

- 수강 클래스 선택 후 결제를 요청한다

applicationContext.xml

```

<!-- Spring Quartz -->
<bean name="runMeJob"
    class="org.springframework.scheduling.quartz.JobDetailFactoryBean"
    p:durability="true"
    >
    <property name="jobClass"
        value="com.power.bean.util.RunMeJob" />
    <property name="jobDataAsMap">
        <map>
            <entry key="LogProcessor" value-ref="LogProcessor" />
        </map>
    </property>
</bean>
<!-- Scheduler Factory -->
<!-- 스케줄러 관리를 위한 Scheduler Factory 를 정의 , job 과 trigger 함께 기술 -->
<bean id = "cronTrigger" class = "org.springframework.scheduling.quartz.CronTriggerFactoryBean">
    <property name = "jobDetail" ref = "runMeJob"/>
    <!-- 매일 오전 0시 실행 -->
    <property name = "cronExpression" value = "0 0 0 * * ?"/>
</bean>

<bean id = "cronTriggerScheduler" class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
    <property name ="triggers">
        <list><ref bean = "cronTrigger"/></list>
    </property>
</bean>

```

RunMeJob.java

```

public class RunMeJob extends QuartzJobBean{

    private LogProcessor logProcessor;

    //실행을 원하는 메소드 호출
    @Override
    protected void executeInternal(JobExecutionContext context) throws JobExecutionException {

        logProcessor.process();
        logProcessor.classFinProcess();
        logProcessor.mailProcess();

    }

    //실제 실행될 테스트를 setter 방식으로 주입
    public void setLogProcessor(LogProcessor logProcessor) {
        this.logProcessor = logProcessor;
    }

}

```

applicationContext.xml

- JobDetailFactoryBean 빈을 이용해 스프링 퀴츠를 사용
- cronTrigger을 이용해 퀴츠가 실행될 시간 지정

RunMeJob.java

- QuartzJobBean을 상속받음
- 퀴츠 실행 시 실행할 메소드를 명시

<기능 설명>

- 매일 0시에 퀴츠를 실행해 기한이 완료된 수업의 상태를 F 로 바꿈
- 수업 기간이 만료된 날, 수강하던 학생들에게 종강 메일을 전송

applicationContext.xml

```
<!-- 메일 전송 -->
<bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl">
  <property name="host" value="smtp.gmail.com" />
  <property name="port" value="587" />
  <property name="username" value="temp59382" />
  <property name="password" value="temp1234%" />
  <property name="javaMailProperties">
    <props>
      <prop key="mail.transport.protocol">smtp</prop>
      <prop key="mail.smtp.auth">true</prop>
      <prop key="mail.smtp.starttls.enable">true</prop>
      <prop key="mail.debug">true</prop>
    </props>
  </property>
</bean>
```

LogProcessor.java

```
public List<String> classFinStudentClassInform(List<String> studentClassInform) {
    List<String> classFinStudentClassInformList = new ArrayList<String>();
    JSONParser parser = new JSONParser();
    for (String informString : studentClassInform) {
        String jsonString = "{" + informString + "}";
        Object obj;
        try {
            obj = parser.parse(jsonString);
            JSONObject jsonObj = (JSONObject) obj;
            Iterator keyIterator = jsonObj.keySet().iterator();

            while(keyIterator.hasNext()) {

                String stringkey = keyIterator.next().toString();
                int key = Integer.parseInt(stringkey);
                int classkey = Integer.parseInt((String) jsonObj.get(stringkey));

                LoginDto memberDto = sqlSession.selectOne( MEMBERNAMESPACE+"selectOneMember", key);
                ClassDto classDto = sqlSession.selectOne(CLASSNAMESPACE + "selectOneClass", classkey);
                String className = classDto.getClass_name();
                String email = memberDto.getMember_email();
                String appendString = "\"" + className + "\":\"" + email + "\"";

                classFinStudentClassInformList.add(appendString);

            }

        } catch (ParseException e) {
            e.printStackTrace();
        }
    }

    return classFinStudentClassInformList;
}
```

☆ test mail 내용입니다

보낸사람 VIP <temp59382@gmail.com>

받는사람 <qodbwls70@naver.com>

test mail 입니다

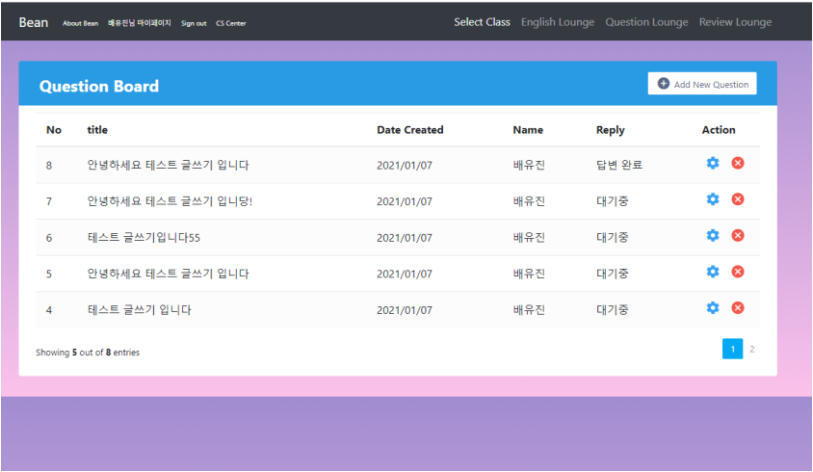
applicationContext.xml

- smtp 관련 bean 생성
- 메일을 보낼 계정의 정보 명시 (google 계정)

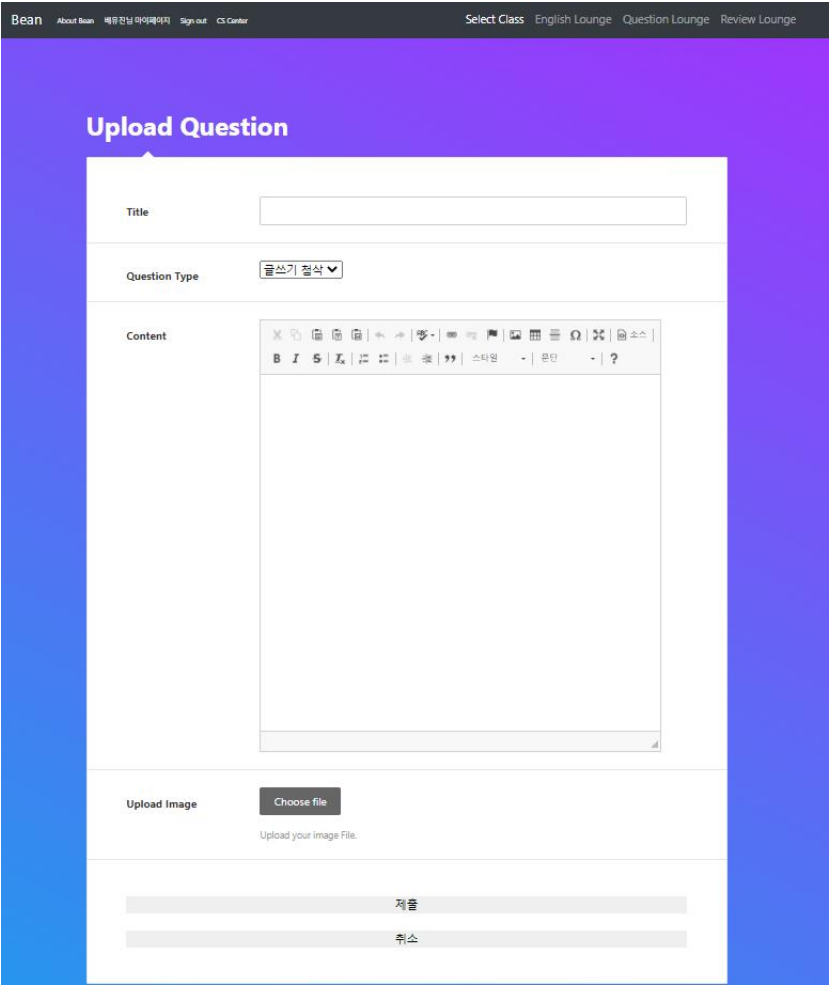
LogProcessor.java

- 0시 전에 끝난 수업의 리스트를 들고 옴
- 수업 리스트에 json 형식으로 저장되어 있는 값을 학생/이메일 정보를 파싱
- 파싱 한 정보를 이용해 메일 전송

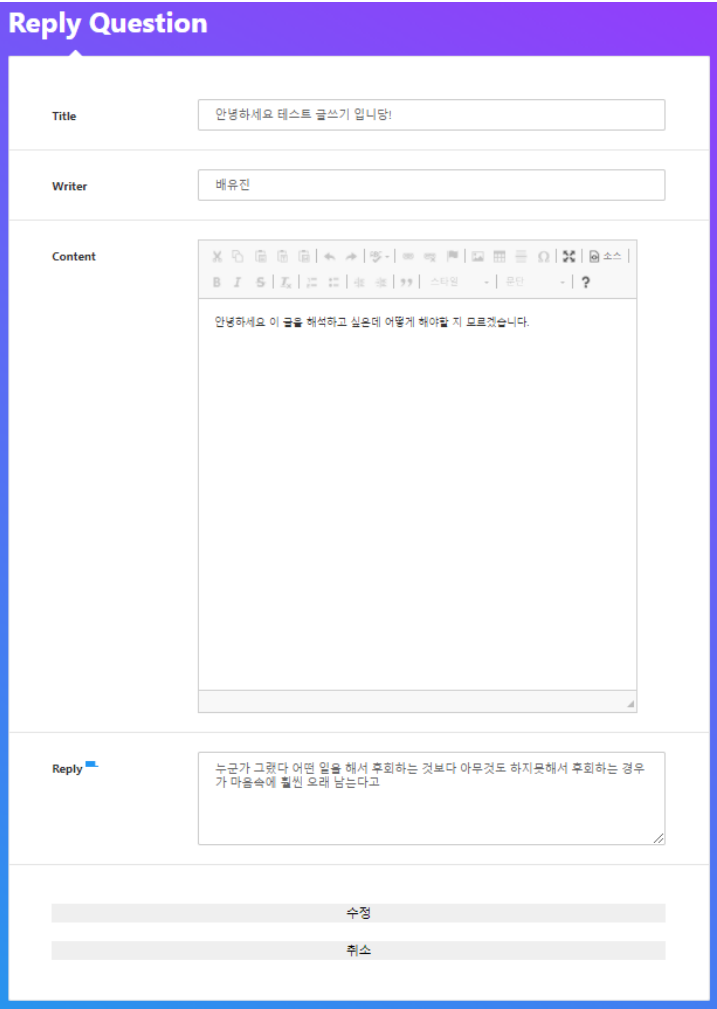
질문 게시판



- 질문 List 를 확인할 수 있다



- 질문을 업로드 할 수 있다



- 선생님 계정의 경우 질문에 답변이 가능하며 토글 버튼으로 ocr 값 사용 여부를 결정할 수 있다

question_upload.xml

```
<form:form method="post" enctype="multipart/form-data"
  modelAttribute="QuestionDto" action="questionUploadres.do">
  <div class="formboard-row">
    <div class="name">Upload Image</div>
    <div class="value">
      <div class="input-group js-input-file">
        <input class="input-file" type="file" name="question_mpfile"
          id="file">
        <label class="Label--file" for="file">Choose
          file</label>
      </div>
      <div class="Label--desc">Upload your image File.</div>
    </div>
  </div>
</div>
```

QuestionController.java

```
// OCR(Connect flask)
Map<String, Object> params = new HashMap<String, Object>();
params.put("path", path);
params.put("filename", name);

try {
  body = objectMapper.writeValueAsString(params);
} catch (JsonGenerationException e) {
  e.printStackTrace();
} catch (JsonMappingException e) {
  e.printStackTrace();
} catch (IOException e) {
  e.printStackTrace();
}

if (body != null) {
  HttpHeaders headers = new HttpHeaders();
  headers.setContentType(new MediaType("application", "json", Charset.forName("UTF-8")));
  HttpEntity entity = new HttpEntity(body, headers);

  ResponseEntity<String> response = restTemplate.postForEntity(url, entity, String.class);
  ocr = response.getBody();
  uploadDto.setQuestionboard_ocr(ocr);
}
```

preprocess_ocr.py

```
@app.route('/', methods = ['POST'])
def image_info():

    #body parse
    params = json.loads(request.data, encoding='utf-8')
    filename=params['filename']
    path = params['path']
    filepath = path + filename
    image = cv2.imread(filepath)

    append_string = image_main(filepath, APP_KEY)

    return append_string
```

question_upload.xml

- mutipart/form-data 를 이용
해 질문 및 사진 전송

QuestionController.java

- 사진이 있을 경우 preprocess_ocr에
사진 관련 데이터 전송
- RestTemplate을 이용해 controller에
서 flask로 통신할 수 있도록 함
- flask 서버에서 ocr 작업 후 추출된
string 값을 json에 담아 반환 받음
- 질문 관련 데이터 DB 저장

preprocess_ocr.py

- 전송받은 이미지 관련 데이터를 토대로
이미지 전처리 진행
- 전처리 한 이미지에 ocr 진행
- ocr string 값을 반환

- 질문 및 질문에 대한 사진을 업로드
- 사진 업로드 시 ocr 진행 후 추출한 값을 질문 답변 시 사용할 수 있도록 함

```
#이미지 이진화
def adaptive_thresholding(image_path:str):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    #Adaptive Thresholding
    max_output_value = 255
    neighborhood_size = 99
    subtract_from_mean = 10

    image_binarized = cv2.adaptiveThreshold(
        image,
        max_output_value,
        cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
        cv2.THRESH_BINARY,
        neighborhood_size,
        subtract_from_mean
    )

    plt.imshow(image_binarized, cmap = 'gray')
    plt.show()

    image_path = "{}_adaptive.PNG".format(image_path)
    cv2.imwrite(image_path, image)

    return image_path
```



```
def kakao_ocr(image_path: str, appkey: str):

    API_URL = 'https://dapi.kakao.com/v2/vision/text/ocr'

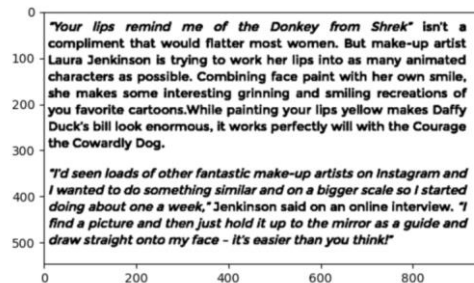
    headers = {'Authorization': 'KakaoAK {}'.format(appkey)}

    image = cv2.imread(image_path)
    jpeg_image = cv2.imencode(".PNG", image)[1]
    data = jpeg_image.tobytes()

    return requests.post(API_URL, headers=headers, files={"image": data})
```

"Your lips remind me of the Donkey from Shrek" isn't a compliment that would flatter most women. But make-up artist Laura Jenkinson is trying to work her lips into as many animated characters as possible. Combining face paint with her own smile, she makes some interesting grinning and smiling recreations of you favorite cartoons. While painting your lips yellow makes Daffy Duck's bill look enormous, it works perfectly will with the Courage the Cowardly Dog.

"I'd seen loads of other fantastic make-up artists on Instagram and I wanted to do something similar and on a bigger scale so I started doing about one a week," Jenkinson said on an online interview. "I find a picture and then just hold it up to the mirror as a guide and draw straight onto my face - it's easier than you think!"



preprocess_ocr.py

- 이미지 선명도 높이기, 이진화, 이미지 대비 높이기 중 ocr처리에 가장 높은 성능을 발휘한 이진화 이용
- 전송받은 값에 대한 전처리 진행 (resize, 이미지 이진화)
- kakao 이미지를 이용해 글자 추출 후 반환

<느낀점>

restTemplate은 빠르지만 connectionPool을 따로 만들어야 여러 곳에서 함께 사용할 수 있다. 이를 추후 보완해야겠다 생각했다

번역 기능

Bean

About Beanteacher강사님 마이페이지Sign outCS Center

Select ClassEnglish LoungeQuestion LoungeReview Lounge

TransLator

Before

영어

After

한국어

번역

- 번역 기능을 사용할 수 있는 페이지이다

TranslateController.java

```
String clientId = "클라이언트 아이디값";
String clientSecret = "시크릿 값";
String apiURL = "https://openapi.naver.com/v1/papago/n2mt"; // API 서버주소

try {
    text = URLEncoder.encode(text, "UTF-8"); // 변환할 문장을 UTF-8로 인코딩 합니다.
    Map<String, String> requestHeaders = new HashMap<>();
    requestHeaders.put("X-Naver-Client-Id", clientId);
    requestHeaders.put("X-Naver-Client-Secret", clientSecret);

    String responseBody = post(apiURL, requestHeaders, source, target, text);

    HttpURLConnection con = connect(apiUrl);
    // 파파고 API 서버로 전달할 파라미터를 설정
    String postParams = "source=" + source + "&target=" + target + "&text=" + text;
    try {
        con.setRequestMethod("POST");
        for (Map.Entry<String, String> header : requestHeaders.entrySet()) {
            con.setRequestProperty(header.getKey(), header.getValue());
        }
    }
```

```
// JSON 형식으로 출력된 결과값을 파싱하는 과정!
JSONParser parser = new JSONParser();
Object ob = (JSONObject) parser.parse(responseBody.toString());
JSONObject jsonObj = (JSONObject) ob;

Object obje = jsonObj.get("message");
JSONObject jsonObj2 = (JSONObject) obje;

Object obj = (Object) jsonObj2.get("result");
JSONObject jsonObj3 = (JSONObject) obj;

String res = (String) jsonObj3.get("translatedText");
System.out.println(res);

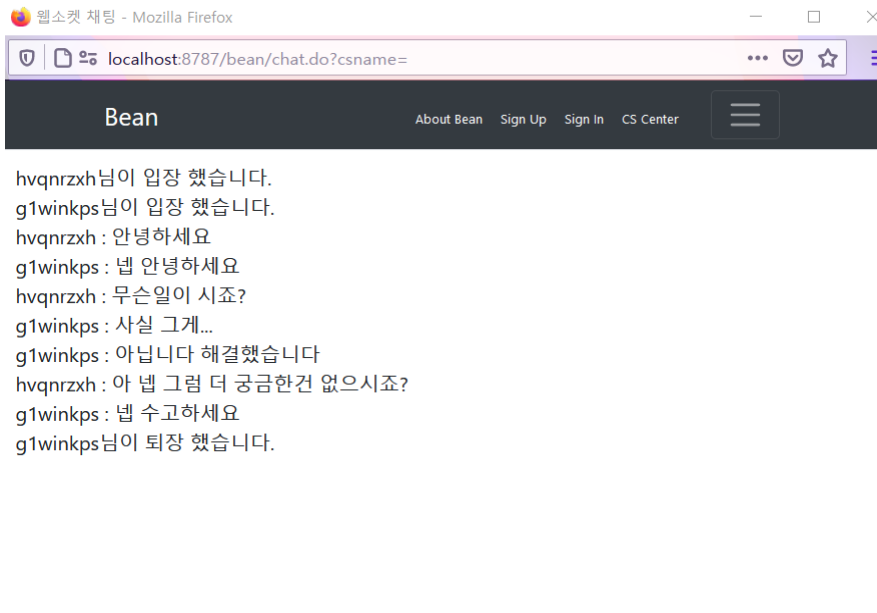
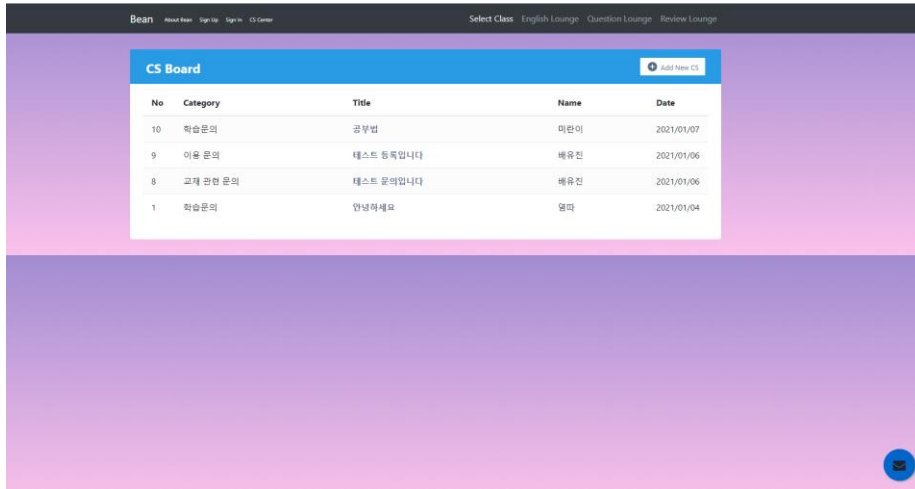
return res;
```

```
{
  "message": {
    "@type": "response",
    "@service": "naverservice.nmt.proxy",
    "@version": "1.0.0",
    "result": {
      "srcLangType": "ko",
      "tarLangType": "en",
      "translatedText": "tea"
    }
  }
}
```

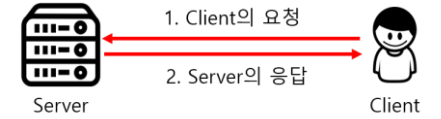
TranslateController.java

- Papago API 를 이용해 번역을 진행
- 번역된 값은 json으로 반환되며, 데이터를 파싱하여 결과값을 화면에 전송함

실시간 채팅

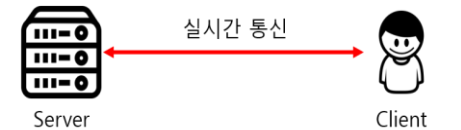


- cs 페이지에서 admin 계정과 채팅할 수 있다



Http 통신

- Client의 요청(Request)이 있을 때만 서버가 응답(Response)하여 해당 정보를 전송하고 곧바로 연결을 종료하는 방식
- Client가 요청을 보내는 경우에만 Server가 응답하는 단방향 통신
- Server로부터 응답을 받은 후에는 연결이 바로 종료
- 실시간 연결이 아니고, 필요한 경우에만 Server로 요청을 보내고 싶을 때 유용



Socket 통신

- Server와 Client가 특정 Port를 통해 실시간으로 양방향 통신을 하는 방식
- Server와 Client가 계속 연결을 유지하는 양방향 통신
- Server와 Client가 실시간으로 데이터를 주고받는 상황이 필요한 경우에 사용

- 채팅에 적합한 socket 통신 이용

ws-config.xml

```
<!-- websocket handler -->
<websocket:handlers allowed-origins="*">
  <websocket:mapping handler="echoHandler" path="/echo"/>
  <websocket:sockjs websocket-enabled="true"/>
</websocket:handlers>

<bean id="echoHandler" class="com.power.bean.util.EchoHandler"/>
```

TextWebSocketHandler.java

```
// 클라이언트가 서버로 연결 처리
@Override
public void afterConnectionEstablished(WebSocketSession session) throws Exception {

    // 채팅방에 접속한 사용자 세션을 리스트에 저장
    sessionList.add(session);

    for (WebSocketSession websocketSession : sessionList) {

        websocketSession.sendMessage(new TextMessage(session.getId() + "님이 입장 했습니다.~"));

    }
}

// 클라이언트가 서버로 메시지 전송 처리
@Override
protected void handleTextMessage(WebSocketSession session, TextMessage message) throws Exception {

    for (WebSocketSession websocketSession : sessionList) {

        websocketSession.sendMessage(new TextMessage(session.getId() + ":" + message.getPayload()));

    }

}

// 클라이언트가 연결을 끊음 처리
@Override
public void afterConnectionClosed(WebSocketSession session, CloseStatus status) throws Exception {

    // 채팅방에서 퇴장한 사용자 세션을 리스트에서 제거
    sessionList.remove(session);

    // 모든 세션에 채팅 전달
    for (int i = 0; i < sessionList.size(); i++) {

        WebSocketSession s = sessionList.get(i);
        s.sendMessage(new TextMessage(session.getId() + "님이 퇴장 했습니다.~"));

    }
}
```

pom.xml

- spring-websocket, Jackson-databind 를 dependency에 추가
- websocket은 spring 4.0 기반이기 때문에 spring 4.3.22 이용

ws-config.xml

- url에서 /echo라는 요청이 오면 com.power.bean.util.EchoHandler에서 처리

TextWebSocketHandler.java

- hadlerTextMessage : 메시지를 보낼 때 override 된 메소드가 동작
- afterConnectionClosed : 접속을 끊었을 때 override 된 메소드가 동작

cschat.js

```

var websocket = {
  init : function(param) {
    this._url = param.url;
    this._initSocket();
  },
  sendChat : function() {
    this._sendMessage($('#message').val());
    $('#message').val("");
  },
  receiveMessage : function(str) {
    $('#divChatData').append('<div>' + str + '</div>');
  },
  closeMessage : function(str) {
    $('#divChatData').append('<div>' + '연결 끊김 : ' + str + '</div>');
  },

  disconnect : function() {
    this._socket.close();
  },
  _initSocket : function() {
    this._socket = new SockJS(this._url);
    this._socket.onmessage = function(evt) {
      websocket.receiveMessage(evt.data);
    };
    this._socket.onclose = function(evt) {
      websocket.closeMessage(evt.data);
    }
  },
  _sendMessage : function(str) {
    this._socket.send(str);
  }
}

```

```

$(document).ready(function() {
  websocket.init({
    url : '<c:url value="/echo" />'
  });
});

```

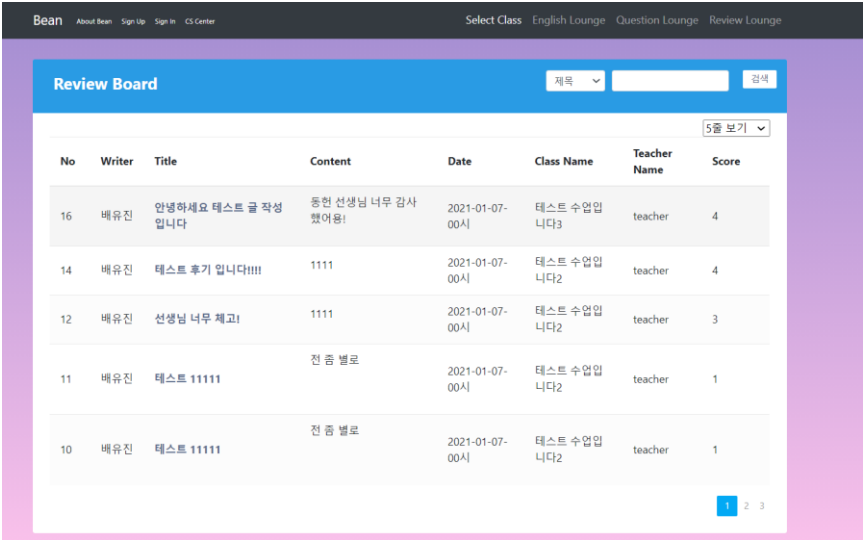
cschat.js

- 소켓 통신 이용
- onmessage : 메시지가 도착하면 호출
- onerror : 에러가 발생하면 호출
- onclose : 웹 소켓이 닫히면 호출
- send : 메시지 전송
- close : 웹 소켓 닫기

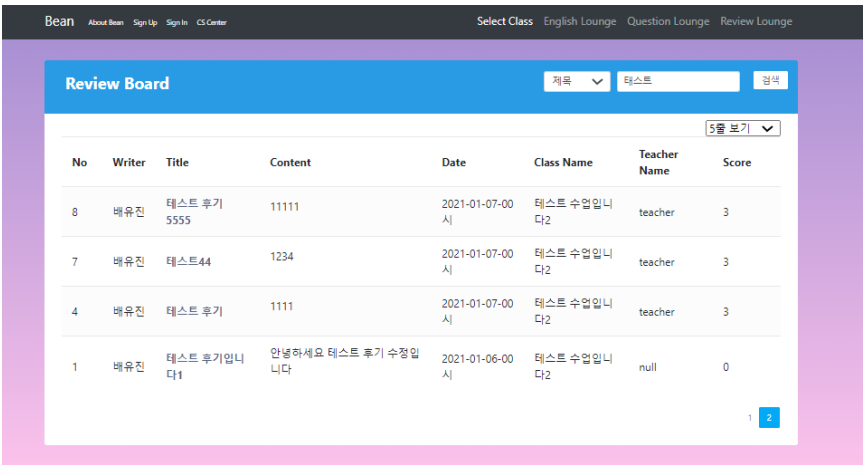
<느낀점>

웹소켓이라는 새로운 내용을 해본 건 정말 좋았다. 다음에는 세부적인 기능을 추가할 예정이다

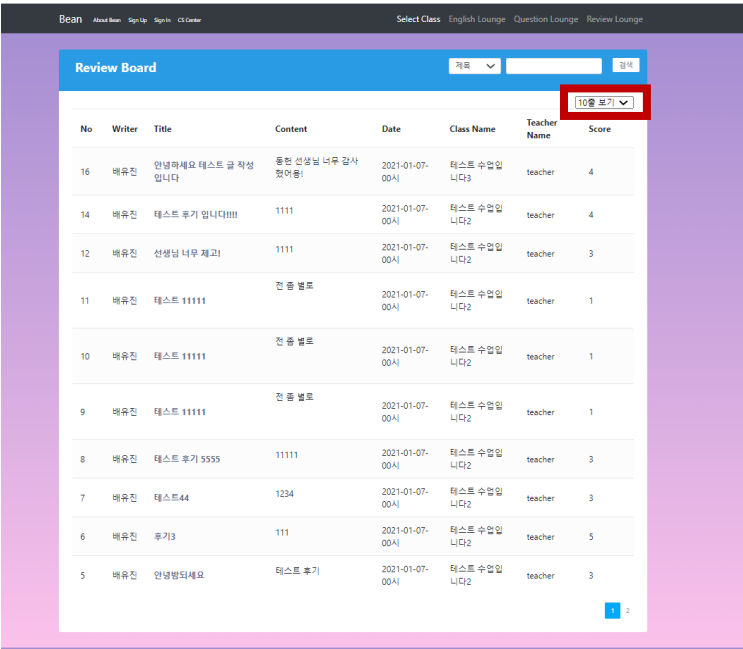
리뷰 게시판(페이징/검색)



-리뷰 게시판 및 다른 게시판 모두 페이징 처리를 했다



- 검색 시에도 페이징 처리가 되도록 했다



- 검색 후에도 줄 수를 바꿀 수 있으며, 줄 수를 바꿨을 때도 페이징 처리가 되도록 했다

review_list.jsp

```
@RequestMapping("/review_list.do")
public String selectList(Model model, PagingDto dto,
    @RequestParam(value = "nowPage", required = false) String nowPage,
    @RequestParam(value = "cntPerPage", required = false) String cntPerPage, String searchType,
    String keyword) {

    Map<String, Object> map = new HashMap<String, Object>();
    Gson gson = new GsonBuilder().create();
    String searchjson;

    map.put("searchType", searchType);
    map.put("keyword", keyword);

    searchjson = gson.toJson(map);

    int total = 0;

    if (nowPage == null && cntPerPage == null) {
        nowPage = "1";
        cntPerPage = "5";
    } else if (nowPage == null) {
        nowPage = "1";
    } else if (cntPerPage == null) {
        cntPerPage = "5";
    }

    total = biz.countSearchCount(searchType, keyword);
    dto = new PagingDto(total, Integer.parseInt(nowPage), Integer.parseInt(cntPerPage));

    model.addAttribute("paging", dto);
    model.addAttribute("list", biz.selectPagingReview(dto, searchType, keyword));
    model.addAttribute("searchType", searchType);
    model.addAttribute("keyword", keyword);
    model.addAttribute("searchjson", searchjson);

    return "review_list";
}
```

```
function selChange() {

    var searchjson = ${searchjson};
    var sel = document.getElementById('cntPerPage').value;

    if(searchjson != null){
        var searchType = searchjson['searchType'];
        var keyword = searchjson['keyword'];

        location.href = "/bean/review_list.do?nowPage=${paging.nowPage}&cntPerPage="+ sel + "&searchType=" + searchType + "&keyword=" + keyword;
    }else{
        location.href = "/bean/review_list.do?nowPage=${paging.nowPage}&cntPerPage="+ sel;
    }
}

<div style="float: right;">
    <select id="cntPerPage" name="sel" onchange="selChange()">
        <option value="5">
            <c:if test="${paging.cntPerPage == 5}">selected</c:if>>5<
        </option>
        <option value="10">
            <c:if test="${paging.cntPerPage == 10}">selected</c:if>>10<
        </option>
        <option value="15">
            <c:if test="${paging.cntPerPage == 15}">selected</c:if>>15<
        </option>
        <option value="20">
            <c:if test="${paging.cntPerPage == 20}">selected</c:if>>20<
        </option>
    </select>
</div>
```

review_list.jsp

- 검색 시에도 페이징 처리가 되도록 설정
- 줄 수를 변경했을 때도 페이징 처리가 유지되도록 설정
- 모든 게시판에 페이징 처리를 진행함

review_mapper.xml

```
<select id="selectList" resultType="ReviewDto" parameterType="hashmap">
    SELECT *
    FROM (
        SELECT ROWNUM
        RN, A.* FROM ( SELECT * FROM REVIEWBOARD
        <trim prefix="WHERE" prefixOverrides="AND/OR">
            <if test="keyword != null and keyword != ''">
                <if test="searchType=='reviewboard_title'">
                    AND REVIEWBOARD_TITLE like '%|#{keyword}|'
                </if>
                <if test="searchType=='reviewboard_content'">
                    AND REVIEWBOARD_CONTENT like '%|#{keyword}|'
                </if>
                <if test="searchType=='reviewboard_name'">
                    AND REVIEWBOARD_NAME like '%|#{keyword}|'
                </if>
            </if>
        </trim>
        ORDER BY REVIEWBOARD_NO DESC) A
    )
    WHERE RN BETWEEN #{start} AND #{end}
</select>
```

감사합니다